



Welcome United States Patent and Trademark Office

[Home](#) | [Login](#) | [Logout](#) | [Access Information](#) | [Alerts](#) | [Sitemap](#) | [Help](#)[Search Session History](#)[BROWSE](#)[SEARCH](#)[IEEE XPLORE GUIDE](#)[SUPPORT](#)

Wed, 22 Jun 2005, 12:18:22 PM EST

Edit an existing query or
compose a new query in the
Search Query Display.

Search Query Display

Select a search number (#)
to:

- Add a query to the Search Query Display

- Combine search queries using AND, OR, or NOT

- Delete a search

- Run a search

Recent Search Queries

Results

#1	(hilton r.<in>au)	5
#2	(hilton r.<in>au)	5
#3	(ronald hilton<in>metadata)	0
#4	(instruction set simulator<in>metadata)	56
#5	(instruction set simulation<in>metadata)	14
#6	(emulation<in>metadata) <and> (instruction<in>metadata) &... <i>legacy</i>	1
#7	(self modifying code<in>metadata) <and> (emulation<in>meta...	0
#8	(instruction<in>metadata) <and> (translat<in>metadata)	0
#9	(instruction<in>metadata) <and> (translation<in>metadata) ...	134
#10	((instruction<in>metadata) <and> (translation<in>metadata) ... <i>emulation</i>	5

[Help](#) [Contact Us](#) [Privacy & Security](#) [IEEE.org](#)

PORTAL

10/27/02

THE ACM DIGITAL LIBRARY

SEARCH

Full Text Search

Advanced Search

Try this search in The ACM Guide

Search by relevance

expanded from

Save results to a binder

Search tips

Open results in a new window

Results 41 - 60 of 200

Result page: PREVIOUS 1 2 3 4 5 6 7 8 9 10 NEXT

Best 200 shown

References 0/0

41 An instruction set and microarchitecture for instruction level distributed processing
 Peter G. Sasse, David H. Ackey, Stephanie Forrest, Darko Stefanovic
 Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems
 May 2005

An instruction set architecture (ISA) suitable for future microprocessor design constraints is proposed. The ISA has hierarchical register files with a small number of accumulators at the top. The instruction set is designed to support a small number of instructions (operands) where inter-operand dependencies are passed through the accumulator. The general register file is divided into two parts: a small number of strands and for holding global values that have many consumers. A microarchitecture to support...

42 Randomized instruction set emulation
 Peter G. Sasse, David H. Ackey, Stephanie Forrest, Darko Stefanovic
 Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems
 May 2005

Injecting binary code into a running program is a common form of attack. Most defenses employ a technique called instruction set emulation (ISE). ISE is a technique where the instructions of a program are passed through an application's machine code. If foreign binary code is injected into a program running under ISE, it will not be executable because it will not know the proper randomization. The paper...

Keywords: Automated diversity, randomized instruction sets, software diversity

43 Hardware supported optimization: Static strands, safety collapsing dependence chains for increasing embedded power efficiency
 Peter G. Sasse, D. Scott Willis, Gabriel H. Loh
 Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems
 May 2005

Modern embedded processors are designed to maximize execution efficiency--the amount of performance achieved per unit of energy dissipated while meeting minimum performance levels. To increase this efficiency, we propose utilizing static strands, dependence chains without fan-out which are emulated by a compiler pass. These dependent instructions are rescheduled to be sequential and are emulated to communicate their location to the hardware. Importantly, this modified application is binary compatible...

Keywords: architecture, dependency collapsing, embedded, energy, sequentially

44 tcc: a system for fast, flexible, and high-level dynamic code generation
 Peter G. Sasse, David H. Ackey, R. Frans Kaasboer
 Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems
 May 2005

tcc is a compiler that provides efficient and high-level access to dynamic code generation. It implements the C (TCC-C) programming language, an extension of ANSI C that supports dynamic code generation [15]. C gives power and flexibility in specifying dynamically generated code: whereas most other

45 TraceBack: first fault diagnosis by reconstruction of distributed control flow
 Andrew Ayers, Richard Schoeder, Chris Metcal, Anant Agarwal, Junghwan Rhee, Emmett Witchel
 ACM SIGPLAN Notices, Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation, Volume 40 Issue 6
 May 2005

Faults that occur in production systems are the most important faults to fix, but most production systems lack the debugging facilities present in development environments. TraceBack provides debugging information for production systems by providing execution history data about program problems (such as crashes, hangs, and exceptions). TraceBack supports features commonly found in production environments such as multiple threads, dynamically loaded modules, multiple source languages (e.g., Java)...

Keywords: fault diagnosis, instrumentation

46 Compilation and run-time systems: Vacuum packing: extracting hardware-detected program phases for possible optimization
 Peter G. Sasse, David H. Ackey, Matthew C. Merten, Wen-mei W. Hwu
 Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture
 November 2002

This paper presents Vacuum Packing, a new approach to profile-based program optimization. Instead of using traditional aggregate or summarized execution profile weights, this approach uses a transparent hardware profiler to automatically detect execution phases and record branch profile information for each new phase. The code extraction algorithm then produces code packages that are specially formed for their corresponding phases. The algorithm compensates for the incomplete and often incoherent...

47 The Performance of Runtime Data Cache Prefetching in a Dynamic Optimization System
 Jiwel Lu, Howard Chen, Rao Fu, Wei-Chung Hsu, Bobbie Chinn, Pen-Ching Yew, Dong-Yuan Chen
 Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture
 December 2003

Traditional software controlled data cache prefetching is often ineffective due to the lack of runtime cache miss and miss address information. To overcome this limitation, we implement runtime data cache prefetching in the dynamic optimization system ADOPRE (Adaptive Object code RE-optimization). Its performance has been compared with static software prefetching on the SPEC2000 benchmark suite. Runtime cache prefetching shows better performance. On an Itanium 2 based Linux workstation, it can increase...

48 Supporting autonomic computing functionality via dynamic operating system kernel aspects
 Michael Engel, Bernd Fraisleben
 Proceedings of the 4th international conference on Aspect-oriented software development
 March 2003

To master the complexity of software systems in the presence of unexpected events potentially affecting system operation, autonomic computing (AC) aims to build systems that have the ability to control and reconfigure themselves. The basic principles employed by autonomic computing are self-configuration, self-optimization, self-healing and self-protection. Typically, these principles are cross-cutting concerns, so...

49 Dynamo: a transparent dynamic optimization system
 Vasanth Balaj, Evelyn Duesterwald, Sanjeev Banerjee
 ACM SIGPLAN Notices, Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation, Volume 35 Issue 5
 May 2000

We describe the design and implementation of Dynamo, a software dynamic optimization system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream to Dynamo can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in the latter, more challenging situation, in order to emphasize the...

- 50** Slim binarizes
Michael Franz, Thomas Kister
December 1997 *Communications of the ACM*, Volume 40 Issue 12
Full text available: [PDF \(242.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 51** Using the SimOS machine simulator to study complex computer systems
Michael Rosenblum, Edward Beggs, Scott Devine, Stephen A. Herrod
April 2000 *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, Volume 7 Issue 1
Full text available: [PDF \(238.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 52** Overcoming the challenges to feedback-directed optimization (Keynote Talk)
Michael D. Smith
January 2000 *ACM SIGPLAN Notices, Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization*, Volume 35 Issue 2
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 53** Feedback-directed optimization (FDO) is a general term used to describe any technique that alters a program's execution based on tendencies observed in its present or past runs. This paper reviews the current state of affairs in FDO and discusses the challenges inhibiting further acceptance of these techniques. It also argues that current trends in hardware and software technology have resulted in an execution environment where immutable executables and traditional static optimizations are ...
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 54** Binary translation
Richard L. Sites, Anton Chrenoff, Matthew B. Kirk, Maurice P. Marks, Scott G. Robinson
February 1999 *Communications of the ACM*, Volume 36 Issue 2
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 55** Exploring Code Cache Eviction Granularities in Dynamic Optimization Systems
Kim Hazelwood, James E. Smith
November 2000 *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization*
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 56** Dynamic optimization systems store optimized or translated code in a software-managed code cache in order to minimize reuse of transformed code. Code caches store superblocks that are not fixed in size, may contain links to other superblocks, and carry a high replacement overhead. These additional constraints reduce the effectiveness of conventional hardware-based cache management policies. In this paper, we explore code cache management policies that evict large blocks of code from the code cache, thus ...
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 57** DCGs: an efficient, relocatable dynamic code generation system
Dawson R. Engler, Todd A. Proebsting
November 1994 *Proceedings of the sixth international conference on Architectural support for programming languages and operating systems*, Volume 29, 28 Issue 11, 5
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 58** Dynamic code generation allows aggressive optimization through the use of runtime information. Previous systems typically relied on ad hoc code generators that were not designed for relocatability, and did not shield the client from machine-specific details. We present a system, dco, that allows clients to generate code for a target machine without knowledge of the target's details. Our one-pass code generator is easily reconfigured and extremely efficient (code generation costs approx ...
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 59** Instruction path code generators
Vijay Gnanu, John Paul Shen
May 2000 *ACM SIGARCH Computer Architecture News, Proceedings of the 27th annual international symposium on Computer architecture*, Volume 29 Issue 2
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)

- 60** This paper presents the concept of an Instruction Path Coprocessor (I-COP), which is a programmable on-chip coprocessor, with its own mini-instruction set, that operates on the core processor's instructions on a per-instruction basis. The I-COP can be more efficiently executed. It is located off the chip, but its execution is controlled by the core processor. The I-COP can be used to improve the core processor's cycle time or pipeline depth. An I-COP is highly versatile and can be used ...
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 61** DISC: a programmable macro engine for customizing applications
Marc L. Coniss, E. Christopher Lewis, Amir Roth
May 2000 *ACM SIGARCH Computer Architecture News, Proceedings of the 20th annual international symposium on Computer architecture*, Volume 29 Issue 2
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 62** Dynamic Instruction Stream Editing (DISC) is a cooperative software-hardware scheme for efficiently adding customization functionality--e.g. safety/security checking, profiling, dynamic code decomposition, and dynamic optimization--to an application. In DISC, application customization functions (ACFs) are formulated as rules for macro-expanding certain instructions into parameterized instruction sequences. The processor executes the rules on the fetched instructions, feeding the result ...
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 63** Adaptive code unloading for resource-constrained JVMs
Unpil Zhang, Chandra Kratz
April 2004 *ACM SIGPLAN Notices, Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*, Volume 39 Issue 7
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 64** Complete-only JVMs for resource-constrained embedded systems have the potential for using device resources more efficiently than interpreter-only systems since compilers can produce significantly higher quality code and code can be stored and reused for future invocations. However, this additional storage requirement for reuse of native code bodies, introduces memory overhead not imposed in interpreter-based systems. In this paper, we present a Java Virtual Machine (JVM) extension for adaptive code ...
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 65** Identifying and Exploiting Spatial Regularity in Data Memory References
Tushar Noman, Brons R. de Supinski, Sally A. Hickey, Frank Mueller, Andy Yoo, Martin Schulz
November 2000 *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 66** The growing processor/memory performance gap causes the performance of many codes to be limited by memory accesses. If known to exist in an application, strided memory accesses forming streams can be targeted by optimizations such as prefetching, relocation, remapping, and vector loads. Unfortunately, many of these optimizations require knowledge of the memory stream's stride. Existing stream-detection mechanisms either require special hardware, which may not gather statistics for subsequent analysis, or are limited ...
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 67** Contributed papers: The donkey strikes back: extending the dynamic interpretation "constructively."
Jim Fernandez
September 2000 *Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics*
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)
- 68** The dynamic interpretation of formula as a binary relation (inducting transitions) on states is extended by alternative treatments of implication, universal quantification, negation and disjunction that are more powerful than the standard ones. The new approach is based on a new logic, called "donkey logic" (which, nonetheless, can be recovered from the new connectives). An analysis of the "donkey" sentence followed by the assertion "It will kick back" is provided.
Full text available: [PDF \(252.12 KB\)](#)
Address information: [Ed. address](#), [abstract](#), [indexing](#), [links](#), [backlinks](#)

Results 41 - 60 of 200 Result page: [previous](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery Copyright © 2005 ACM, Inc.
[Terms of Use](#) [Privacy Policy](#) [Contact Us](#)
 Useful downloads: [Adobe Acrobat](#) [QuickTime](#) [Windows Media Player](#) [Real Player](#)

- Previous research has shown that the SPEC benchmarks achieve low miss ratios in relatively small instruction caches. This paper presents evidence that current software-development practices produce applications that exhibit substantially higher instruction-cache miss ratios than do the SPEC benchmarks. To represent these trends, we have assembled a collection of applications, called the Instruction Benchmark Suite (IBS), that provides a better test of instruction-cache performance. We discuss th ...

- Keywords:** SMP, SPARC vs ISA, UltraSPARC, complete machine simulator, execution-driven simulation, object-oriented design


- 14 Machine-adaptable dynamic binary translation**
David Ung, Cristina Chioresu
January 2000
ACM SIGPLAN Notices, 'Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization', Volume 35 Issue 7
Full text available at citeseer.berkeley.edu
Additional information: ML@CS.DUKE.EDU,

- Dynamic translation** techniques have normally been limited to two particular machines; a competitor's machine and the hardware manufacturer's machine. This research provides for a more general framework for dynamic translation, by providing a framework based on specifications of machines that...
- Keywords:** binary translation, dynamic compilation, dynamic execution, emulation, interpretation

- Keywords:** computer architecture, computer simulation, computer system performance analysis, operating systems

- ¹⁴ Dynamic translation: Dynamic binary translation for accumulator-oriented architectures
Ho-Seop Kim, James E. Smith
March 2003
Proceedings of the international symposium on Code generation and optimization:
feedback-directed and runtime optimization
P479
P480
P481
P482
P483
P484
P485
P486
P487
P488
P489
P490
P491
P492
P493
P494
P495
P496
P497
P498
P499
P500
P501
P502
P503
P504
P505
P506
P507
P508
P509
P510
P511
P512
P513
P514
P515
P516
P517
P518
P519
P520
P521
P522
P523
P524
P525
P526
P527
P528
P529
P530
P531
P532
P533
P534
P535
P536
P537
P538
P539
P540
P541
P542
P543
P544
P545
P546
P547
P548
P549
P550
P551
P552
P553
P554
P555
P556
P557
P558
P559
P560
P561
P562
P563
P564
P565
P566
P567
P568
P569
P570
P571
P572
P573
P574
P575
P576
P577
P578
P579
P580
P581
P582
P583
P584
P585
P586
P587
P588
P589
P590
P591
P592
P593
P594
P595
P596
P597
P598
P599
P600
P601
P602
P603
P604
P605
P606
P607
P608
P609
P610
P611
P612
P613
P614
P615
P616
P617
P618
P619
P620
P621
P622
P623
P624
P625
P626
P627
P628
P629
P630
P631
P632
P633
P634
P635
P636
P637
P638
P639
P640
P641
P642
P643
P644
P645
P646
P647
P648
P649
P650
P651
P652
P653
P654
P655
P656
P657
P658
P659
P660
P661
P662
P663
P664
P665
P666
P667
P668
P669
P670
P671
P672
P673
P674
P675
P676
P677
P678
P679
P680
P681
P682
P683
P684
P685
P686
P687
P688
P689
P690
P691
P692
P693
P694
P695
P696
P697
P698
P699
P700
P701
P702
P703
P704
P705
P706
P707
P708
P709
P710
P711
P712
P713
P714
P715
P716
P717
P718
P719
P720
P721
P722
P723
P724
P725
P726
P727
P728
P729
P730
P731
P732
P733
P734
P735
P736
P737
P738
P739
P740
P741
P742
P743
P744
P745
P746
P747
P748
P749
P750
P751
P752
P753
P754
P755
P756
P757
P758
P759
P760
P761
P762
P763
P764
P765
P766
P767
P768
P769
P770
P771
P772
P773
P774
P775
P776
P777
P778
P779
P780
P781
P782
P783
P784
P785
P786
P787
P788
P789
P790
P791
P792
P793
P794
P795
P796
P797
P798
P799
P800
P801
P802
P803
P804
P805
P806
P807
P808
P809
P810
P811
P812
P813
P814
P815
P816
P817
P818
P819
P820
P821
P822
P823
P824
P825
P826
P827
P828
P829
P830
P831
P832
P833
P834
P835
P836
P837
P838
P839
P840
P841
P842
P843
P844
P845
P846
P847
P848
P849
P850
P851
P852
P853
P854
P855
P856
P857
P858
P859
P860
P861
P862
P863
P864
P865
P866
P867
P868
P869
P870
P871
P872
P873
P874
P875
P876
P877
P878
P879
P880
P881
P882
P883
P884
P885
P886
P887
P888
P889
P890
P891
P892
P893
P894
P895
P896
P897
P898
P899
P900
P901
P902
P903
P904
P905
P906
P907
P908
P909
P910
P911
P912
P913
P914
P915
P916
P917
P918
P919
P920
P921
P922
P923
P924
P925
P926
P927
P928
P929
P930
P931
P932
P933
P934
P935
P936
P937
P938
P939
P940
P941
P942
P943
P944
P945
P946
P947
P948
P949
P950
P951
P952
P953
P954
P955
P956
P957
P958
P959
P960
P961
P962
P963
P964
P965
P966
P967
P968
P969
P970
P971
P972
P973
P974
P975
P976
P977
P978
P979
P980
P981
P982
P983
P984
P985
P986
P987
P988
P989
P990
P991
P992
P993
P994
P995
P996
P997
P998
P999
P1000
P1001
P1002
P1003
P1004
P1005
P1006
P1007
P1008
P1009
P1010
P1011
P1012
P1013
P1014
P1015
P1016
P1017
P1018
P1019
P1020
P1021
P1022
P1023
P1024
P1025
P1026
P1027
P1028
P1029
P1030
P1031
P1032
P1033
P1034
P1035
P1036
P1037
P1038
P1039
P1040
P1041
P1042
P1043
P1044
P1045
P1046
P1047
P1048
P1049
P1050
P1051
P1052
P1053
P1054
P1055
P1056
P1057
P1058
P1059
P1060
P1061
P1062
P1063
P1064
P1065
P1066
P1067
P1068
P1069
P1070
P1071
P1072
P1073
P1074
P1075
P1076
P1077
P1078
P1079
P1080
P1081
P1082
P1083
P1084
P1085
P1086
P1087
P1088
P1089
P1090
P1091
P1092
P1093
P1094
P1095
P1096
P1097
P1098
P1099
P1100
P1101
P1102
P1103
P1104
P1105
P1106
P1107
P1108
P1109
P1110
P1111
P1112
P1113
P1114
P1115
P1116
P1117
P1118
P1119
P1120
P1121
P1122
P1123
P1124
P1125
P1126
P1127
P1128
P1129

- A dynamic binary translation system for a co-designed virtual machine is described and evaluated. The underlying hardware directly executes an accumulator-oriented instruction set that exposes instruction-dependence chains (strands) to a distributed microarchitecture containing a simple instruction pipeline. To support conventional program binaries, a source instruction set (Alpha in our study) is dynamically translated to the target accumulator instruction set. The binary translator identifies ...

- An instruction set and microarchitecture for instruction level distributed processing
Ho-Seop Kim, James E. Smith
May 2002
ACM SIGARCH Computer Architecture News, Volume 30 Issue 2
Pp 165-174
doi:10.1145/561421.561424
-  SIGARCH 2002
- Address: Department of Computer Science, University of Illinois at Urbana-Champaign, 201 North Goodwin Avenue, Urbana, IL 61801-2401, USA
Email: smith@uiuc.edu
- An instruction set architecture (ISA) suitable for future microprocessor design constraints is proposed. The ISA has hierarchical register files with a small number of accumulators at the top. The instruction stream is divided into chains of dependent instructions (strands) where intra-strand dependencies are passed through the accumulator. The general-purpose register file is used for communication between strands and for holding global values that have many consumers. A microarchitecture to support ...



- 14** Intrusion detection: Randomized instruction set emulation to disrupt binary code injection attacks
Elena Gabriela Barrantes, David H. Ackley, Trek S. Palmer, Darko Stefanovic, Dima Dai Zovi
October 2003
Proceedings of the 10th ACM conference on Computer and communications security

- Keywords:** automated diversity, emulation, information hiding, language randomization, obfuscation, security

- Emulation of large systems**
S. G. Tucker
December 1985
Published worldwide
 0254-2628/86
Communications of the ACM, Volume 8 Issue 12
Additional information: 24 columns, 67 pages, 6 figures, 1 table

- 20** Binary translation and architecture convergence issues for IBM_system/390
Michael Gschwind, Kemal Ercioğlu, Erik Almman, Sumesh Sathya
May 2005
Proceedings of the 14th International conference on Supercomputing
FIA 0407 00000K
10.1007/s11464-005-1410-2
Address information: LA 3-0701, DALLAS, TEXAS 75263, USA, EICSA

- We describe the design issues in an implementation of the ESA/330 architecture based on binary translation to a very long instruction word (VLIW) processor. During binary translation, complex ESA/330 instructions are decomposed into instruction "primitives" which are then scheduled onto wide-issue machine. The aim is to achieve high instruction level parallelism due to the increased scheduling and optimization opportunities which can be exploited by binary translation software ...

- 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next
 The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.
 Terms of Usage Privacy Policy Code of Ethics Contact Us
 Useful downloads:  Adobe Acrobat  QuickTime  Microsoft Media Player  Real Player

<http://portal.acm.org/results.cfm?coll=ACM&dl=ACM&CFID=48065529&CFTOKEN=3...> 6/22/2005

<http://portal.acm.org/results.cfm?coll=ACM&dl=ACM&CFID=48065529&CFTOKEN=3...> 6/22/2005

10. **Programmable architectures, Dynamic reconfiguration, with binary translation, breaking the ILP barrier, with software compatibility**
Antonio Carlos S. Beck, Luigi Corno
May 2000
Proceedings of the 42nd annual conference on Design automation
P. 1041-1046
P. 1041-1046
Abstract Information: [MAGNET, APLN, INTERACT, JETA, JETA](#)
 11. In this paper, we present the impact of dynamically translating any sequence of instructions into combinational logic. The proposed approach combines a reconfigurable architecture with a binary translation mechanism, being totally transparent for the software designer. Besides ensuring software compatibility, the technique allows porting the same code for different machines tracking technological evolution. The target processor is a Java machine able to execute Java bytecodes. Experimental results ...
Keywords: binary translation, Java, power consumption, reconfigurable processors
 12. **Going native, Module-aware translation for real-life desktop applications**
Jianhui Li, Peng Zhang, Omer Etzion
May 2000
Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments
P. 125-134, 125
P. 125-134, 125
Abstract Information: [MAGNET, APLN, INTERACT, JETA, JETA](#)
 13. A dynamic binary translator is a just-in-time compiler that translates source architecture binaries into target architecture binaries on the fly. It enables the fast running of the source architecture binaries on the target architecture. Traditional dynamic binary translators invalidate their translations when a module is unloaded, so later re-loading of the same module will lead to a full retranslation. Moreover, most of the loading and unloading are performed on a few "hot" modules, which cause ...
Keywords: dynamic binary translation, dynamic loaded module, memory management, translation reuse
 14. **Optimising hot paths in a dynamic binary translator**
David Uno, Cristina Chiment
May 2001
ACM SIGARCH Computer Architecture News, Volume 29 Issue 1
P. 25-32, 25
P. 25-32, 25
Abstract Information: [MAGNET, APLN, INTERACT, JETA, JETA](#)
 15. In dynamic binary translation, code is translated, on the fly, at run-time, while the user perceives ordinary execution of the program on the target machine. Code fragments that are frequently executed follow the same sequence of flow control over a period of time. These fragments form a hot path and are optimized to improve the overall performance of the program. Multiple hot paths may also exist in programs. A program may choose to execute in one hot path for some time, but later switch to another ...
Keywords: binary translation, dynamic compilation, dynamic execution, run-time profiling
 16. **Binary translation and architecture convergence issues for IBM system/390**
Michael Gschwind, Kemal Ercelik, Erik Altman, Suresh Sabharwal
May 2000
Proceedings of the 14th international conference on Supercomputing
P. 145-154, 145
P. 145-154, 145
Abstract Information: [MAGNET, APLN, INTERACT, JETA, JETA](#)
 17. We describe the design issues in an implementation of the ESA/390 architecture based on binary translation to a very long instruction word (VLIW) processor. During binary translation, complex ESA/390 instructions are decomposed into instruction "primitives" which are then scheduled onto a wide-issue machine. The aim is to achieve high instruction level parallelism due to the increased scheduling and optimization opportunities which can be exploited by binary translation software. ...
 18. **Using Event-Based Translation to Support Dynamic Protocol Evolution**
Nathan D. Ryan, Alexander L. Wolf
May 2000
Proceedings of the 26th International Conference on Software Engineering
P. 147-156, 147
P. 147-156, 147
Abstract Information: [MAGNET, APLN, INTERACT, JETA, JETA](#)
 19. All systems built from distributed components involve those of one or more protocols for inter-component communication. Whether these protocols are based on a broad-based "standard" or are specially designed for a particular application, they are likely to evolve. The goal of the workdescribed here is to contribute techniques that can support protocol evolution. We are concerned not with how or why a protocol might evolve, or even whether that evolution fits some sense correct. Rather, our concern ...

- Mathias Neubaier, Michael Seifert
October 2001
ACM SIGPLAN Notices, *Proceedings of the sixth ACM SIGPLAN international conference on Functional programming*, Volume 36 Issue 10
94
Address: neubaier, seifert, pfaffenberger@informatik.uni-leipzig.de

It is possible to translate code written in Emacs Lisp or another Lisp dialect which uses dynamic scoping to a more modern programming language with lexical scoping while largely preserving structure and control flow. This is achieved by translating lexical closures to first-class objects. The translation of dynamic binding to suitable instances of lexical environments in Emacs Lisp is the basis for a more general translation of dynamic binding to lexical environments. This translation of programs in fact exhibit identical behavior under both dynamic and lexical binding. At ...
 - Dynamic typing for distributed programming in polymorphic languages**
Oguzhan Arslan
January 1999
100
Address: arslan@cs.cmu.edu

While static typing is widely accepted as being necessary for secure program execution, dynamic typing is more popular for the development of distributed programming environments. Dynamics have been proposed as a language construct for dynamic environments with languages such as CLI, Cedar/Mesa, and Modula-3. However proposals for incorporating dynamic typing into languages with parametric polymorphism have serious shortcom ...

Keywords: dynamic typing, marshalling, parametric polymorphism, static typing
 - DAISY: dynamic compilation for 100% architectural compatibility**
Kenan Ercioğlu, Erik R. Altman
May 1997
187
Address: erciglu@cs.cmu.edu, altman@cs.cmu.edu

ACM SIGARCH Computer Architecture News, *Proceedings of the 24th annual international symposium on Computer architecture*, Volume 25 Issue 2
10
Address: erciglu@cs.cmu.edu, altman@cs.cmu.edu

Although VLIW architectures offer the advantages of simplicity of design and high issue rates, a major impediment to their use is that they are not compatible with the existing software base. We describe new simple hardware features for a VLIW machine we call DAISY (Dynamically Architected Instruction Set from *hardware*). DAISY is specifically intended to emulate existing architectures, so that all existing software ...

Keywords: binary translation, dynamic compilation, instruction-level parallelism, object code compatible VLIW, superscalar
 - Safe polymorphic type inference for a dynamically typed language: translating Scheme to ML**
Fritz Heintz, Jakob Rehof
October 1999
190
Address: heintz@informatik.uni-leipzig.de, rehof@informatik.uni-leipzig.de

Proceedings of the seventh international conference on Functional programming languages and computer architecture
9
Address: heintz@informatik.uni-leipzig.de, rehof@informatik.uni-leipzig.de

Optimization and precise exceptions in dynamic compilation
 - Michael Gschwind, Erik Altman
March 2001
1
Address: gschwind@cs.cmu.edu, altman@cs.cmu.edu

ACM SIGARCH Computer Architecture News, Volume 29 Issue 1
2
Address: gschwind@cs.cmu.edu, altman@cs.cmu.edu

Maintaining precise exceptions is an important aspect of achieving full compatibility with a legacy architecture. While asynchronous exceptions can be deferred to an appropriate boundary in the code, synchronous exceptions must be taken when they occur. This introduces uncertainty into liveness analysis since processor state that is otherwise dead may be exposed when an exception handler is invoked. Previous systems either had to sacrifice full compatibility to achieve more freedom to perform op ...
 - Word rendering and a dynamic programming beam search algorithm for statistical machine translation**
Christoph Tillmann, Hermann Ney
March 2000
1
Address: tillmann@informatik.uni-leipzig.de, ney@informatik.uni-leipzig.de

In this article, we describe an efficient beam search algorithm for statistical machine translation based on dynamic programming (DP). The search algorithm uses the translation model presented in Brown et al. (1993). Starting from a CP-based solution to the traveling-salesman problem, we present a novel technique to restrict the search algorithm. Word renderings restrictions especially useful for the tr ...

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery, Copyright © 2005 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:



[Adobe Acrobat](#)



[QuickTime](#)



[Windows Media Player](#)



[Real Player](#)